How to Control Processes with Large Deadtimes

Large deadtimes present challenges for controllers. Here are some practical strategies.

John Gerry, ExperTune

Processes with large deadtimes present a special challenge for a controller—any controller. The controller must wait until the deadtime has passed before it gets any feedback from the process.

The best thing to do for controlling this type of process is to try to reduce the dead time. Simply moving a probe closer to the valve may do it. But sometimes there is nothing that can be done to the process. What then?

Picking and tuning

The control algorithm for a deadtime process is actually very simple. A PI controller does a fine job.

Other options for controllers include model-predictive; model-based; pole-cancellation; internal model control; dynamic-matrix controller; Dahlin; or a direct synthesis type. They all use a model inside the controller and could be called model-based controllers. These controllers can seek out slightly better response than PI—but sacrifice robustness.

The robustness penalty using model-based control (made to be even the same speed as a PI) is severe—process deadtime can go up or down and the loop will go unstable. Increasing deadtime for any process will eventually create instability. But decreasing deadtime to create instability is weirdly counter-intuitive. If the deadtime were to decrease with a PI controller, the loop would remain stable. So why not keep it simple and just use a PI controller?

The optimal tuning of a PI controller for a deadtime only process is simple:

- Controller gain = 0.3/(process gain);
- Integral time = 0.5 × (process dead time)

This tuning gives minimum error to a step-function upset in the process. The integral units are measured in minutes or seconds. For a slower response, simply lower the gain a little. (These equations apply for usual ideal and series PI algorithms. For a parallel algorithm, divide the integral time by controller gain.)

Additional lag

If the process has a small lag in addition to the deadtime, "cancel" the lag by setting the derivative time equal to it. This works if the lag is about five times less than the deadtime.

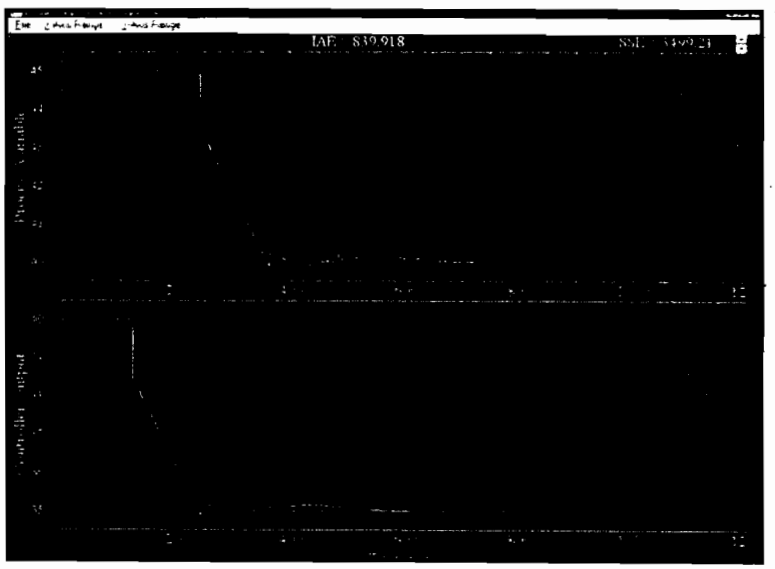

Fig. 1: Strategies vary in dealing with a 4-hr deadtime.

CONTROL ENGINEERING

KEY WORDS

- Process control & instrumentation
- Loop controllers
- Loop-tuning software
- PID (Proportional/Integral/Derivative)

In the example shown in the first screen capture (Fig. 1), a process has a gain of one and deadtime of 120 min. It is controlled with a PI controller (red line) having an integral time of 60 min, and a gain of 0.3. It shows that 240 min or two deadtimes pass before the process reaches setpoint or recovers from an upset.

Filtering the process variable (second-order filter with a time constant of 1/20th the deadtime) will smooth out the sharp edges of the response (shown by the yellow line).

Adaptive deadtime

A powerful adaptive deadtime controller can be made if deadtime can be measured, or it's possible to infer deadtime from a variable like the speed of a machine or flow.

To make the controller adaptive, simply adjust the integral action in the controller using the above formula. Measuring or inferring the deadtime is the key.

This kind of adaptive controller is much more robust than using an identification technique that

Tuning trade offs: fast response or robustness

depends on normal process noise. Attempting to identify deadtime using a least-squares fit or some other modeling method that looks at normal process noise can be subject to gross modeling errors with disturbances. Identification techniques relying on setpoint changes, however, are accurate.

Model predictive

In a model-predictive controller the user sets a desired speed of response—the first order response time to setpoint desired.

The goal of the model predictive controller is to match the setpoint response to a first order time constant (or lag time) entered. Response is first delayed by the process deadtime. The blue line (in

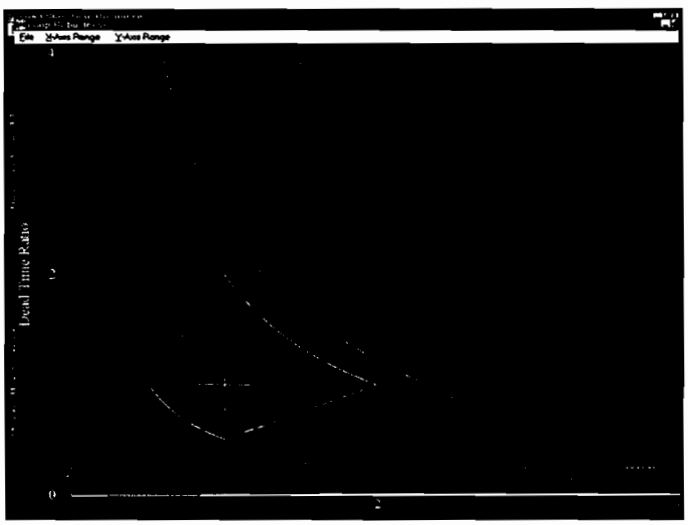

Fig. 2: At the cross, within the stable region, the gain and delay ratios are equal to one, and process gain and deadtime are at the tuned process values.

Fig. 1) shows the response using a model-predictive controller with a speed of response adjusted to be the similar robustness as the PI.

Robustness is the other side of the story. There is always a trade off between fast response and robustness or sensitivity to process changes. Robustness plots easily show this tradeoff. They show how sensitive (or robust) the loop is to process gain or process deadtime changes. The plot has two axes:

- Gain ratio = (process gain)/(process gain to which the controller was tuned);
- Delay ratio = (process deadtime)/(process deadtime to which the controller was tuned).

In the second screen capture (Fig. 2), the plot has regions of stability and instability. The red and blue lines on the robustness plot are the limit of stability for PI and model- predictive controllers. To the right and above the solid lines (higher gain and delay ratios) the closed-loop process is unstable. To the left and below the solid lines, the closed-loop system is stable. The cross represents the point where both the gain and delay ratios are equal to one. In other words, the assumed process deadtime and the actual deadtime are identical at the cross, as are the assumed process gain and actual process gain.

Generally, a safety factor or divisor of 2 is "reasonable" for a loop. These points are represented by the vertices of the blue line "box" in the robustness plot. It is a design aid. To achieve practical system stability, the limit of stability line should be outside the "box." The vertices are connected by lines that are straight on a log-log plot.

In this example, for either PI or model-predictive controller, if the process gain were to increase by a factor of 2.3, the loop would be unstable. Start at the white cross and move horizontally to the right. Where the stability lines intersect, the gain ratio is about 2.3.

Another example that works for either controller is if the deadtime were to increase by 2.4. Starting at the cross and moving straight up into the stability lines, the deadtime ratio is about 2.4. At this point the loop would again be at the verge of stability.

Counter-intuitive result

A third example that applies to the model- predictive controller is valid only if the process deadtime were to decrease. Again start at the cross and move straight down until smashing into the blue line. At this point the deadtime ratio is about 0.5. The loop with the model-predictive controller goes unstable with deadtime *decrease*.

Intuitively, like all controllers, the model- predictive becomes unstable with a large enough increase in the process deadtime. But, counter to intuition and the behavior of a PI, it also goes unstable when the process deadtime falls.

A PI controller is simple, familiar, and does a great job controlling processes with large deadtimes—simply set the gain and integral time as a factor of the process gain and deadtime. Although a model-predictive controller may perform slightly better, it suffers from poor robustness and requires extensive training for operations personnel. □

John Gerry, P.E., ceo and president of ExperTune (Hubertus, Wis.), has an MS Degree in Chemical Engineering, with experience at Eastman Kodak, Eli Lilly, S.C. Johnson, and The Foxboro Co. Simulations and figures were created using the ExperTune Loop Simulator.

For more information, Circle 225 or visit www.controleng.com/info.

Was this article...

Interesting? 393 Useful? 394 Not useful? 395

Comments?
Send an e-mail to mhoske@cahners.com.